WHAT IS CLAIMED IS:

1. A method of co-simulating a digital circuit using a simulation engine which communicates with at least one first programming language by means of a foreign language interface and which communicates directly with at least one second programming or hardware description language, comprising the steps of:

(a) providing at least one first model of at least one first part of the digital circuit in at least one high-level hardware description language which supports concurrent processes communicating with each other;

(b) converting the at least one first model to at least one software model in the at least one first language;

(c) providing at least one second model of at least one second part of the digital circuit in the at least one second language; and

(d) applying the at least one software model in the at least one first language and the at least one second

model in the at least one second language to the simulation engine.

2. A method as claimed in claim 1, in which the converting step (b) comprises compiling the at least one first model in the at least one high-level hardware description language to the at least one software model in the at least one first language.

3. A method as claimed in claim 1, in which the at least one high-level hardware description language is based on a communicating sequential processes model.

4. A method as claimed in claim 1, in which the at least one first part of the digital circuit is represented in the at least one high-level hardware description language as a plurality of concurrent processes which communicate with each other and the converting step (b) comprises converting the concurrent processes to a sequential software process.

5. A method as claimed in claim 4, in which the software process comprises at least one stimulus unit for detecting a predetermined stimulus and at least one response unit

for providing a predetermined response in response to the at least one stimulus unit.

6. A method as claimed in claim 5, in which at least one of the response units comprises a process response unit for performing a desired behaviour of the at least one first part of the digital circuit.

7. A method as claimed in claim 6, in which the desired behaviour comprises a plurality of discrete processes triggered by a common event and the process response unit comprises a scheduler for scheduling the discrete processes and a process handler for performing the discrete processes in accordance with the scheduling.

8. A method as claimed in claim 7, in which the scheduler: forms a list of active unhandled processes having respective exit points; chooses from the list a current process; and selects an entry point for the current process.

9. A method as claimed in claim 8, in which, at each exit point, the scheduler chooses from the list a further current process and selects a further entry point for

the further current process.

10. A method as claimed in claim 7, in which the converting step (b) comprises: generating, for at least one of the discrete processes, software code including a program loop having a jump instruction and a loop termination condition; analysing the loop termination condition to determine whether it is possibly non-terminating; and, if so, replacing the jump instruction with an exit point.

11. A method as claimed in claim 8, in which the converting step (b) comprises: generating, for at least one of the discrete processes, software code including a program loop having a jump instruction and a loop termination condition; analysing the loop termination condition to determine whether it is possibly nonterminating; and, if so, replacing the jump instruction with an exit point, and

in which, at the exit point, the scheduler places the at least one discrete process in the list of active unhandled processes with a new entry point.

12. A method as claimed in claim 9, in which the converting step (b) comprises: generating, for at least one of the

discrete processes, software code including a program loop having a jump instruction and a loop termination condition; analysing the loop termination condition to determine whether it is possibly nonterminating; and, if so, replacing the jump instruction with an exit point, and

in which, at the exit point, the scheduler places the at least one discrete process in the list of active unhandled processes with a new entry point.

13. A method of designing a digital circuit, comprising performing a method as claimed in claim 1, checking whether the result of the co-simulation is correct, checking whether the digital circuit is synthesisable, and generating a low-level hardware description of the digital circuit.

14. A method of manufacturing a digital circuit, comprising performing a method as claimed in claim 13 and forming, from the low-level hardware description, an integrated circuit including the digital circuit.

15. An integrated circuit made by a method as claimed in claim 14.

16. An apparatus for performing a method as claimed in claim 1.

17. An apparatus as claimed in claim 16, comprising a computer programmed by a computer program.

18. A computer program for an apparatus as claimed in claim 17.

19. A storage medium containing a program as claimed in claim 18.